NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

FAST SAMPLING OF GAUSSIAN MARKOV RANDOM FIELDS WITH APPLICATIONS

by

Håvard Rue

PREPRINT
STATISTICS NO. 1/2000

NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
TRONDHEIM, NORWAY

This report has URL http://www.math.ntnu.no/preprint/statistics/2000/S1-2000.ps
Håvard Rue has homepage: http://www.math.ntnu.no/~hrue
E-mail: hrue@math.ntnu.no
Address: Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491
Trondheim, Norway.

Fast Sampling of Gaussian Markov Random Fields with Applications

HÅVARD RUE
DEPARTMENT OF MATHEMATICAL SCIENCES
NTNU, NORWAY

JANUARY 6, 2000

SUMMARY

This paper demonstrate how Gaussian Markov random fields (conditional autoregressions) can be fast sampled using numerical techniques for sparse matrices. The algorithm is general, surprisingly efficient, and expands easily to various forms for conditional simulation and evaluation of normalisation constants. I demonstrate its use in Markov chain Monte Carlo algorithms for disease mapping, space varying regression model, spatial non-parametrics, hierarchical space-time modelling and Bayesian imaging.

KEYWORDS: Conditional Autoregressive model, Divide and Conquer, Gaussian Markov random field, Markov chain Monte Carlo, Block-sampling.

ADDRESS FOR CORRESPONDENCE: H. Rue, Department of Mathematical Sciences, The Norwegian University for Science and Technology, N-7491 Trondheim.
E-MAIL: Havard.Rue@math.ntnu.no

ACKNOWLEDGEMENTS: The author thanks Merrilee A. Hurn, Oddvar K. Husby, Leo Knorr-Held, Arne Martinsen, Håkon Tjelmeland and Darren J. Wilkinson for stimulating discussions, and Leo Knorr-Held for providing the oral cavity cancer data and the region-map of Germany. This work has been supported by EU TMR network ERB-FMRX-CT96-0095 on "Computational and Statistical methods for the analysis of spatial data".

1 Introduction

Gaussian Markov random fields (GMRF) or conditional autoregressions, are popular and commonly used (sub-)models within spatial statistics in a broad sense, see for example Cressie (1993), Besag & Kooperberg (1995), Winkler (1995) the references therein, and Kiiveri & Campbell (1989), Besag, York & Mollié (1991), Heikkinen & Arjas (1998), Wikle, Berliner & Cressie (1998) and Assunção, Gamerman & Assunção (1999). GMRFs are convenient models both from a computation and theoretical point of view: they have a Markov-property and are jointly Gaussian. The Markov property is also important for models relying on Markov chain Monte Carlo (MCMC) based inference, as it will ensure rapid computation of the conditional densities.

Sampling from a GMRF is straight-forward in theory. A GMRF is multivariate Gaussian hence all well-known general algorithms apply, see for example the review in Cressie (1993). Non of these algorithms take particular advantage of the Markov property for the GMRF, and hence will be slow and computational inconvenient for larger problems. For a GMRF with a circulant block-Toeplitz covariance matrix, sampling is particular efficient using fast Fourier transforms (Wood & Chan, 1994; Dietrich & Newsam, 1997). However, the Toeplitz assumption only applies to the most simple problems. Recently, new algorithms have been constructed that make specific use of the Markov property of the GMRF, by either running a special dynamic model with artificial observed data (Lavine, 1998), or by using the propagation algorithm of Lauritzen (1992) with extensions by Dawid (1992). However, neither of these seems particular useful nor fast unless in simple cases, for example for a GMRF involving the four nearest neighbours on a lattice. For more complicated and realistic neighbourhood patterns, the algorithm complexity increases (very) rapidly both with respect to CPU and coding.

In this paper, it will be demonstrated how to use numerical techniques for sparse matrices to construct an efficient way to sample from a general GMRF defined on a lattice or a graph. The algorithm is simple, surprisingly efficient and expands easily to complicated neighbourhood structures and various forms for conditional simulation and evaluation of the likelihood.

In addition to traditional applications of GMRF in spatial statistics referenced above, there are also new potential uses of GMRF. Rue & Tjelmeland (1999) study how to approximate a stationary Gaussian field to a GMRF (on a lattice) with somewhat surprising results: a Gaussian field with a Gaussian correlation function with range 50 could be approximated with a GMRF using a 7×7 neighbourhood with a maximum error in the correlation function of about 0.01, and similarly with other families for correlation functions. Thus, a local GMRF can in practice replace a conventionally specified Gaussian field in applications, so the posterior analysis or other tasks can make benefit of the new fast algorithms and analytical simplifications for GMRFs.

In spatial models using MCMC for inference, the new algorithms allow for block-sampling or block-proposals depending on whether the full conditional is a GMRF or contains additional non-quadratic terms. The full generality in the algorithms are important, as the conditioning on the "rest" often results in a non-homogeneous GMRF. Block-sampling in MCMC algorithms most often improves the properties of the MCMC algorithm. Previous examples in the literature include dynamic models where the natural structure and dimension of the problems ease the construction and implementation (Carter & Kohn, 1994; Shephard & Pitt, 1997), and expert systems (Jensen, Kong & Kjærulff, 1995). The experiments demonstrate large improvements when the block-sampling is efficient. On the theoretical side, Roberts & Sahu (1997) study the convergence rate for (Gibbs) block-sampling a GMRF.

Rapid evaluation of the likelihood for GMRFs are important for estimating (hyper-)parameters using either a likelihood or Bayesian approach. For example, Griffith & Sone (1995) investigate numerical approximations formulas to the likelihood for which the neighbourhood structure was build upon spatial neighbouring regions, and Kiiveri (1992) approximate numerically the integral in Whittle's approximation (in the stationary case). Our new algorithms can efficiently compute the likelihood for large cases even in the general non-homogeneous case, which makes numerical approximation (even for simpler cases) less important. Note that Rue & Tjelmeland (1999) demonstrate that likelihood based inference for a GMRF seems to be quite sensitive for model error.

The reminder of the paper is organised as follows. Section 2 presents the basic algorithm for sampling from a GMRF. Section 3 discusses extensions including various forms for conditioning and evaluation of the likelihood. Section 4 discuss blocking- and a divide and conquer-strategy to approach large graphs. In Section 5, I present some examples and demonstrate how the algorithms can be used in disease mapping, space varying regression, spatial non-parametrics, hierarchical space-time models and Bayesian imaging, improving the MCMC algorithm used for analysing the model. Computational details are described in the Appendix.

2 THE ALGORITHM

This section defines a GMRF and describe the basic algorithm for producing samples from it.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V}$ is the set of nodes (or vertices) and $\mathcal{E}$ is the set of edges. For simplicity, we number the nodes from 1 to $n = |\mathcal{V}|$. Define ∂_i as the set of neighbours to node i: the set of all nodes adjacent to node i in the graph. A zero mean GMRF $\boldsymbol{x}$ on $\mathcal{G}$ is Gaussian with precision matrix $\boldsymbol{Q}$, such that Q_{ij} is zero iff $j \notin \partial_i \cup i$. A GMRF is often specified through the local characteristics

$$\mathrm{E}(x_i \mid \boldsymbol{x}_{-i}) = -\sum_{j \in \partial_i} \frac{Q_{ij}}{Q_{ii}} x_j, \qquad \text{and} \qquad \mathrm{Var}(x_i \mid \boldsymbol{x}_{-i}) = Q_{ii}^{-1},$$

which illustrate the Markov-property of a GMRF.

The fast sampling algorithm depend on the typical structure of the graph for GMRFs used in spatial statistics. A common application in spatial statistics starts with Figure 1 showing the map of Germany consisting of 544 regions. A common prior for a spatial model of regional data (which is discussed further in Section 5.1 and Section 5.2), is an intrinsic GMRF $\boldsymbol{x}$ where for each region i, ∂_i is the set of all regions j adjacent to i (Besag & Kooperberg, 1995)

$$\pi(\boldsymbol{x} \mid \kappa) \propto \kappa^{n/2} \exp\left(-\frac{1}{2}\kappa \sum_{i \sim j} (x_i - x_j)^2\right). \tag{1}$$

The graph for the GMRF is then derived from the map. Figure 2a shows the non-zero pattern in the corresponding precision matrix using the numbering of the regions as in the definition of the map. (Recall that $Q_{ij} = 0$ iff $i \not\sim j$ and $i \neq j$.) Figure 2b shows the same precision matrix after a permutation of the nodes to minimise the bandwidth. I will comment later on how this is done. The bandwidth for the precision matrix is defined as $b_w = \max_{i \sim j} |i - j|$. The permuted

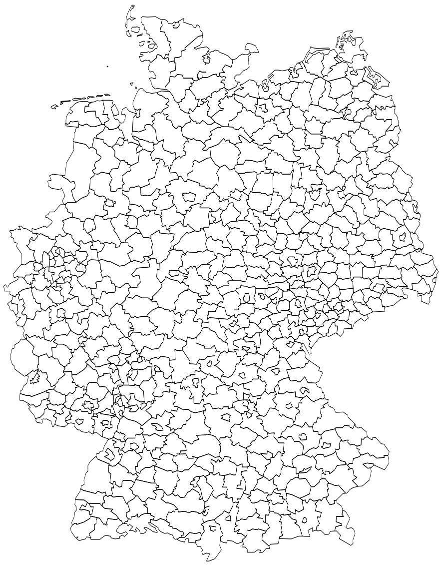

FIGURE 1: The map of Germany with 544 regions.

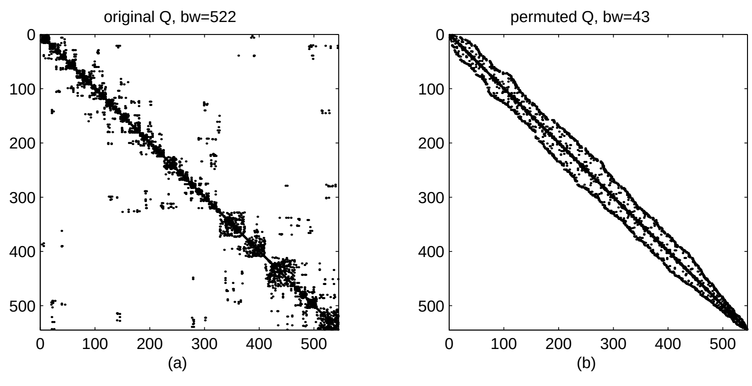

FIGURE 2: Figure (a) show the non-zero entries in the precision matrix $\boldsymbol{Q}$ for the GMRF using the numbering of the regions as in the definition of the map. Figure (b) shows the same precision matrix after a permutation of the nodes to minimise the bandwidth. Fast numerical algorithms exists for band-matrices and these are the basis for our algorithms.

precision matrix is a band-matrix (with a small bandwidth) and the basis for our algorithms. Fast algorithms for most tasks exists for band-matrices and their efficiency depends on b_w. Note that the permutation of the nodes reduce b_w from 522 to 43, which allows for a potentially large speedup.

The situation is similar for lattices where the neighbours ∂_i is the set of sites closest to i using some norm. Assume for simplicity that ∂_i is the $(2m+1) \times (2m+1)$ square centred at i and that the lattice has dimension n with n_1 rows and n_2 columns, where $n_1 \leq n_2$. In this case the precision matrix using row-wise ordering, is also a band-matrix with bandwidth $b_w = m(n_1+1)$. In this case there is no need to permute the nodes, as the solution is obvious.

The banded precision matrix for the GMRF (after permutation) is the basis for our sampling algorithm. There are no additional assumptions of the coefficients in the precision matrix. The algorithm goes as follows: First note that computing the Cholesky factorisation of the band matrix $\boldsymbol{Q}$ with bandwidth b_w

$$\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T \tag{2}$$

is a standard problem in numerics. By Theorem 4.3.1 in Golub & van Loan (1996), $\boldsymbol{L}$ has lower bandwidth b_w (and is zero above the diagonal) and is efficiently computed using only nb_w^2 flops, see Golub & van Loan (1996, Sec. 4.3.5) for details and algorithm. $\boldsymbol{L}$ require nb_wU bytes of storage where U is the number of bytes needed to store one float. In the next step let $\boldsymbol{z}$ be a vector of independent standard Gaussians, and note by direct calculation that $\boldsymbol{x}$ defined by the solution of

$$\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{z} \tag{3}$$

has mean zero and precision matrix $\boldsymbol{Q}$. Again, solving (3) efficiently is a standard problem in numerics and the solution can be computed in $2nb_w$ flops using band back-substitution, see again Golub & van Loan (1996, Sec. 4.3.2) for details and algorithm. How to permute the nodes to minimise the bandwidth is also a classic numerical problem, for which there exists clever and fast algorithms. Algorithmic details are described in the Appendix.

Some additional notes to the algorithm are as follows.

REPEATED SAMPLES Sampling more than one sample from the same GMRF is especially efficient. The band Cholesky factorisation in (2) is computed only once and then (3) is solved for each sample.

COMPUTATIONAL COST Assume for simplicity a $n_1 \times n_2$, $n_1 \leq n_2$, lattice with a $(2m+1)\times(2m+1)$ neighbourhood. Computing the band Cholesky decomposition (2) costs $n_2 n_1^3 m^2$ flops, and solving (3) costs $2n_2 n_1^2 m$ flops. Note that the cost is linear in the *largest* dimension n_2, and cubic/quadratic in the *smallest* dimension n_1. Hence, it is far more efficient to sample from rectangular compared to square lattices. A similar comment is also valid for graphs.

A COUNTEREXAMPLE It is not hard to construct a graph for which the bandwidth equals the dimension of the problem, $b_w = n-1$ for all permutations: let every node have all other nodes as neighbours. In this cases, there is little to gain computationally. In spatial applications the bandwidth is most often small compared to the problem size $b_w = o(n)$, and often $b_w = \mathcal{O}(\sqrt{n})$.

3 Conditioning and Likelihood

I will now discuss generalisations of the simulation algorithm to conditional simulation and evaluation of the log-likelihood.

3.1 Conditional Simulation

This section discuss three common forms of conditional sampling for a GMRF, $i)$ conditioning on an observed subset of $\boldsymbol{x}$, $ii)$ conditioning on "the rest" as is common in MCMC algorithms, and $iii)$ conditioning on some linear combinations of $\boldsymbol{x}$.

3.1.1 Sampling from $\boldsymbol{x}_1|\boldsymbol{x}_2$

Assume we want to sample a subset of a zero mean GMRF $\boldsymbol{x}$ conditioned on the rest. Split $\boldsymbol{x}$ into $\boldsymbol{x}_1$ (unobserved) and $\boldsymbol{x}_2$ (observed) where $\boldsymbol{x} = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T)^T$, and divide $\boldsymbol{Q}$ similarly,

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix}.$$

The task is to sample from $\boldsymbol{x}_1|\boldsymbol{x}_2$. This conditional density is Gaussian with mean $\boldsymbol{\mu}_{1|2}$ given by $\boldsymbol{Q}_{11}\boldsymbol{\mu}_{1|2} = -\boldsymbol{Q}_{12}\boldsymbol{x}_2$ and precision $\boldsymbol{Q}_{1|2} = \boldsymbol{Q}_{11}$. These nice and simple expressions for the conditional densities are one of the great advantages using GMRFs. The proof is simply to use the block equations derived from $\boldsymbol{Q}\boldsymbol{Q}^{-1} = \boldsymbol{I}$ and the standard results about conditional mean and variance. Conditioning on observed values are particularly easy for GMRF, since the conditional precision matrix is explicitly known as the precision matrix for the graph after removing all observed nodes. Note further that $\boldsymbol{Q}_{11}$ cannot have larger bandwidth than $\boldsymbol{Q}$, so it is still a band-matrix (after permutation) and the algorithm in Section 2 can be used. The only difference is the need to compute $\boldsymbol{\mu}_{1|2}$ by solving $\boldsymbol{Q}_{11}\boldsymbol{\mu}_{1|2} = \boldsymbol{b}$, where the right hand side is easily computable $\boldsymbol{b} = -\boldsymbol{Q}_{12}\boldsymbol{x}_2$. $\boldsymbol{\mu}_{1|2}$ can be obtained from the band-Cholesky decomposition of $\boldsymbol{Q}_{11} = \boldsymbol{L}\boldsymbol{L}^T$, by solving efficiently by forward (band) substitution $\boldsymbol{L}\boldsymbol{u} = \boldsymbol{b}$ and a back (band) substitution $\boldsymbol{L}^T\boldsymbol{\mu}_{1|2} = \boldsymbol{u}$. To obtain a sample, one only needs to add the solution of $\boldsymbol{L}^T\boldsymbol{y} = \boldsymbol{z}$ (where $\boldsymbol{z}$ is a vector of independent standard Gaussians) to the conditional mean, $\boldsymbol{x}_{1|2} = \boldsymbol{\mu}_{1|2} + \boldsymbol{y}$. The computational cost for a conditioned sample is then $n_1 b_{w1}^2 + 2 \times 2n_1 b_{w1}$ flops, where $n_1 = \dim(\boldsymbol{x}_1)$ and b_{w1} is the bandwidth of the permuted $\boldsymbol{Q}_{11}$ matrix. Repeated samples cost $2n_1 b_{w1}$ flops.

Note that the case $\boldsymbol{x}_1|\boldsymbol{x}_2$ also covers the case when some observations are observed exact and others with Gaussian errors. This is because the conditional density for $\boldsymbol{x}$ conditioned on the noisy observations is again a GMRF with precision matrix $\boldsymbol{Q} + \boldsymbol{Q}^{\text{obs}}$, where $\boldsymbol{Q}^{\text{obs}}$ is the precision matrix for the observations. A common situation is to observe (conditional independent) $y_i|\boldsymbol{x}$ with mean x_i and precision τ_i for $i \in \mathcal{I} \subset \mathcal{V}$, then $Q_{ii}^{\text{obs}} = \tau_i, i \in \mathcal{I}$ and zero otherwise. After computing the conditional mean, one can condition on the exact observations as in the previous paragraph.

3.1.2 SAMPLING FROM $\pi(\boldsymbol{x}) \propto \exp(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x})$

A typical form of the conditional density when deriving full conditionals for GMRF in MCMC algorithms (examples are shown in Section 5), is

$$\pi(\boldsymbol{x} \mid \text{rest}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x}\right). \tag{4}$$

Note that $\boldsymbol{Q}$ is the precision matrix and the mean is implicitly given as $\boldsymbol{Q}^{-1}\boldsymbol{b}$, similar to the case in Section 3.1.1. To sample from (4) I therefore compute the band Cholesky decomposition $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$ and solve the following systems: $\boldsymbol{L}\boldsymbol{v} = \boldsymbol{b}$, $\boldsymbol{L}^T\boldsymbol{\mu} = \boldsymbol{v}$, $\boldsymbol{L}^T\boldsymbol{y} = \boldsymbol{z}$, and then add up to obtain the sample $\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{y}$.

3.1.3 SAMPLING FROM $\boldsymbol{x}|\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$

A more general problem is to sample $\boldsymbol{x}$ conditioned on $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ (or with Gaussian noise added) where $\boldsymbol{A}$ is a $k \times n$ matrix with rank k. A conditional sample can be obtained using that

$$\boldsymbol{x} - \boldsymbol{Q}^{-1}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^T\right)^{-1}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}), \tag{5}$$

where $\boldsymbol{x}$ is an unconditional sample, has the correct conditional density. Eq. (5) is commonly refered to as conditioning using Kriging (Cressie, 1993, Sec. 3.6.2). In (5) one needs to compute $\boldsymbol{V} = \boldsymbol{Q}^{-1}\boldsymbol{A}^T$, which requires solving $\boldsymbol{Q}\boldsymbol{V}_i = (\boldsymbol{A}^T)_i$ for each of the k columns. Since the band Cholesky decomposition is already available as $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$, I solve $\boldsymbol{Q}\boldsymbol{V}_i = (\boldsymbol{A}^T)_i$ by a forward substitution $\boldsymbol{L}\boldsymbol{u} = (\boldsymbol{A}^T)_i$ and a back-substitution $\boldsymbol{L}^T\boldsymbol{V}_i = \boldsymbol{u}$. I compute $\boldsymbol{W} = \boldsymbol{V}(\boldsymbol{A}\boldsymbol{V})^{-1}$ only once, and for each sample one then need to compute $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, pre-multiply with $-\boldsymbol{W}$ and add $\boldsymbol{x}$.

3.2 EVALUATION OF THE LIKELIHOOD

It is important to compute the likelihood for a sample $\boldsymbol{x}$ from a zero mean GMRF. The band-Cholesky factorisation of the (permuted) precision matrix, allows for fast evaluation of the likelihood, as $|\boldsymbol{Q}|^{1/2}$ equals $\prod_i L_{ii}$. Hence, the log-likelihood $l(\boldsymbol{x})$ is

$$l(\boldsymbol{x}) = -\frac{n}{2}\log 2\pi + \sum_i \log(L_{ii}) - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}. \tag{6}$$

If $\boldsymbol{x}$ is simulated by solving $\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{z}$, then $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ equals $\boldsymbol{z}^T\boldsymbol{z}$, hence $l(\boldsymbol{x})$ can be evaluated with no extra costs. The computation of the log-likelihood for the conditional sample in Section 3.1.1 and Section 3.1.2 is similar.

For a sample from $\boldsymbol{x}|\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ as in Section 3.1.3, I have no fast way to evaluate the likelihood. However, in application using MCMC for doing inference for $\boldsymbol{x}$, one only needs the ratio of the likelihood evaluated at two samples $\boldsymbol{x}$ and $\boldsymbol{x}'$. The identity

$$l(\boldsymbol{x}' \mid \boldsymbol{A}\boldsymbol{x}' = \boldsymbol{b}) - l(\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}) \;=\; l(\boldsymbol{x}') - l(\boldsymbol{x})$$

can then be used, see Rue & Husby (1998) for application and proof.

4 Sampling a GMRF on a Large Graph

For a GMRF on a really large graph it will still be inconvenient to sample or evaluate the likelihood. This is both due to the CPU required and the amount of memory needed to store the band-Cholesky factorisation, although they are much related. To give an idea of how large a "large graph" is, my Sun Ultra 2 computer, can at the time of writing handle lattice-cases up to size 256×256 with a 5×5 neighbourhood. Beyond that, the performance drops due to memory-problems.

I will discuss two strategies to sample a GMRF on a large graph. The first approach reduce the computational burden using blocking-strategies, like a Gibbs sampler for sampling, and pseudo-likelihood as an approximation to the likelihood. The second approach reduce the memory-requirement (mainly) by using a divide and conquer strategy to divide the problem into simpler ones using iterative linear solvers for large linear systems. Both strategies can of course be combined.

4.1 Blocking-strategies

To sample a GMRF on a large graph can be approached similarly as MCMC is used to sample complicated non-Gaussian densities. It is easy to construct a (Gibbs) block-sampler for a GMRF, by dividing the graph into subgraphs $\{\mathcal{G}_i\}$ and sample successively from $\pi(\boldsymbol{x}_{\mathcal{G}_i} \mid \boldsymbol{x}_{-\mathcal{G}_i})$ using the algorithm in Section 3.1. The subgraphs do not need to be disjoint, however, $\cup_i \mathcal{G}_i = \mathcal{G}$. To illustrate the computational savings using this strategy, consider a $n_1 \times n_1$ lattice with a $(2m+1) \times (2m+1)$ neighbourhood. Assuming disjoint subgraphs, where each $\mathcal{G}_i$ is a $n_1/k \times n_1/k$ lattice, the relative computational burden is $1/k^2$ for computing all the band-Cholesky factorisations, and $1/k$ doing repeated sampling.

The pseudo-likelihood approach use $\text{PL}(\boldsymbol{x}) = \prod_i \pi(x_i \mid \boldsymbol{x}_{-i})$ as an approximation to the likelihood. It seems more sensible to use a block-version of the pseudo-likelihood: partition the graph into disjoint subgraphs and combine the conditional likelihoods in each subgraph into

$$\text{PLBlock}(\boldsymbol{x}) = \prod_i \pi(\boldsymbol{x}_{\mathcal{G}_i} \mid \boldsymbol{x}_{-\mathcal{G}_i}).$$

$\text{PLBlock}(\boldsymbol{x})$ can easily be obtained by using the algorithm in Section 3.1 to compute the conditional densities, and then use (6) to evaluate each (conditional) likelihood.

The intuitive way to partition a graph, is to divided it into subgraphs with few links inbetween them. In this way the dependency is large within each subgraph and less inbetween the subgraphs. How sensitive the performance is to the blocking structure is unknown, but Roberts & Sahu (1997) gives the convergence rate for block Gibbs-sampling a GMRF which can be computed and compared for small problems.

Note that computing the permutation of the nodes is still feasible even for large graphs, and suggest implicitly ways to divide the graph into subgraphs. (Think of the permuted precision matrix as the precision matrix for an (one-dimensional) autoregressive process of order b_w.) As an example, consider the GMRF defined on the graph induced by the map of Germany in Figure 1, leading to the precision matrix in Figure 2. Assume we want to divide the graph into 4 subgraphs. We can do this in an automatic manner by using the permuted set of nodes: the first subgraph consists of those nodes where the permuted node are in $1, \dots, n/4$, and so on. This (automatic) procedure will then produce subgraphs by dividing (near perfectly) the map into four connected and homogeneous regions.

4.2 Using a Divide and Conquer Strategy

For large graphs, the memory requirements for the band-Cholesky factorisation is nb_w and often $\mathcal{O}(n^{3/2})$. The reason is that the sampling algorithm in Section 2 needs a splitting of the precision matrix $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$. Iterative solvers for large, sparse and positive definite linear systems, can solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ using only $\mathcal{O}(n)$ memory. I will show how we can construct a sampling algorithm based on a divide and conquer strategy (using the Markov property), that mainly based on iterative solving systems like $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$. This strategy can be used for dividing large graphs into smaller ones such that the algorithm in Section 2 applies for each subgraph.

4.2.1 Solving large, sparse and positive definite linear systems

The sampling algorithm in Section 2 and Section 3 use the band-Cholesky factorisation $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$ to sample and solve equations like $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$. The divide and conquer strategy, derives a sampling algorithm that are based on solving equations like $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$ for a large but sparse positive definite $\boldsymbol{Q}$. We can solve such equations using iterative techniques which needs only $\mathcal{O}(n)$ memory. Again, this is a standard-problem in numerics, see for example Golub & van Loan (1996), Saad (1996), van de Velde (1994), and good public code like for example Oppe, Joubert & Kincaid (1988) exists. The basic method is an iterative conjugate gradient algorithm that search for the minimum of $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ in direction orthogonal in $\boldsymbol{Q}$-norm to all previous search directions. Note that the solution of the minimalisation problem is the solution of $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$. Only matrix-vector products like $\boldsymbol{Q}\boldsymbol{v}$ needs to be computed. Additionally, the system is (linearly) transformed to improve the conditioning of $\boldsymbol{Q}$ and hence the convergence rate.

4.2.2 The Divide and Conquer Idea

The divide and conquer idea is best illustrated when the graph is a lattice, and we assume for the moment a zero mean GMRF defined on a $n_1 \times n_1$ lattice with a 3×3 neighbourhood. Assume also that n_1 is odd, $n_1 = 2c + 1$. Generalisations to general graphs are imidiate after the idea is clear.

Let $\boldsymbol{x}_c$ denote the c'th column. Define $\boldsymbol{x}_A = (\boldsymbol{x}_1^T, \dots, \boldsymbol{x}_{c-1}^T)^T$ and $\boldsymbol{x}_B = (\boldsymbol{x}_{c+1}^T, \dots, \boldsymbol{x}_{n_1}^T)^T$, so that $\boldsymbol{x} = (\boldsymbol{x}_A^T, \boldsymbol{x}_c^T, \boldsymbol{x}_B^T)^T$. We split $\boldsymbol{Q}$ similarly

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{AA} & \boldsymbol{Q}_{Ac} & \boldsymbol{0} \\ \boldsymbol{Q}_{cA} & \boldsymbol{Q}_{cc} & \boldsymbol{Q}_{cB} \\ \boldsymbol{0} & \boldsymbol{Q}_{Bc} & \boldsymbol{Q}_{cc} \end{bmatrix}.$$

Note that $\boldsymbol{Q}_{AB}$ is zero since $\boldsymbol{x}_c$ is a separating subset for $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$. Assume for a moment that we can simulate $\boldsymbol{x}_s$ from its marginal density $\pi(\boldsymbol{x}_s)$. (We will later show how to compute $\pi(\boldsymbol{x}_s)$.) A sample from the joint density $\pi(\boldsymbol{x})$ is then completed by sampling $\boldsymbol{x}_A|\boldsymbol{x}_c$ and $\boldsymbol{x}_B|\boldsymbol{x}_c$ from their from the conditional densities. Similar to Section 3.1.1, the conditional precision matrix and mean for $\boldsymbol{x}_A|\boldsymbol{x}_c$ is $\boldsymbol{Q}_{A|c} = \boldsymbol{Q}_{AA}$ and $\boldsymbol{\mu}_{A|c} = \boldsymbol{Q}_{AA}^{-1}\boldsymbol{Q}_{Ac}\boldsymbol{x}_c$, and similar with $\boldsymbol{x}_{B|c}$. Note that both $\boldsymbol{Q}_{A|c}$ and $\boldsymbol{Q}_{B|c}$ have similar nice band structure as $\boldsymbol{Q}$, so we can solve the equations for the conditional means using the iterative methods described in Section 4.2.1. Hence, we have divided our problem into two separate smaller ones of the same type as the original one.

The remaining task is to compute the marginal density for $\boldsymbol{x}_c$. Since only the precision matrix $\boldsymbol{Q}$ is available, we need to compute the covariance matrix $\boldsymbol{\Sigma}_{cc}$ for $\boldsymbol{x}_c$. Starting with $\boldsymbol{Q}\boldsymbol{\Sigma} = \boldsymbol{I}$, we can write out the equations for block-column c in $\boldsymbol{\Sigma}$

$$\boldsymbol{Q} \begin{bmatrix} \boldsymbol{\Sigma}_{1c} \\ \vdots \\ \boldsymbol{\Sigma}_{cc} \\ \vdots \\ \boldsymbol{\Sigma}_{n_1 c} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{I} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}. \tag{7}$$

The right hand side is zero except for a $n_1 \times n_1$ identity matrix $\boldsymbol{I}$. Note that (7) defines n_1 linear systems, each with coefficient matrix $\boldsymbol{Q}$, hence is solved using iterative techniques as described in Section 4.2.1. From the kth solution we can extract column k of $\boldsymbol{\Sigma}_{cc}$. We simulate $\boldsymbol{x}_c$ using the classic procedure by computing the Cholesky decomposition $\boldsymbol{\Sigma}_{cc} = \boldsymbol{L}\boldsymbol{L}^T$, and then compute $\boldsymbol{x}_c = \boldsymbol{L}\boldsymbol{z}$, where $\boldsymbol{z}$ is a $n_1 \times 1$ vector of iid standard Gaussian variates.

4.2.3 Summary

The divide and conquer step in Section 4.2.2 demonstrate how we can split a problem into two (or more in general) smaller ones of the same type, if we know the marginal density of the separating subset. We solve the linear systems using iterative methods (which has no memory requirement) as described in Section 4.2.1. After the divide and conquer step is performed, we continue to apply the divide and conquer step on each subproblem that is "to large", otherwise we switch to the basic algorithm in Section 2.

The generalisation from a lattice to a general graph is immediate. The only problem is how to automaticly extract a (minimal) separating subset which divide the problem into smaller ones of approximate same size. As in Section 4.1, the computed permutation of the nodes are again useful. Those nodes which has a permuted node in the interval $n/2 - b_w/2$ to $n/2 + b_w/2$ (with obvious integer corrections) will be a separating subset. It might not be strictly minimal, but quite close. Anyway, we can easily correct for this if needed.

5 Applications

In this section I will demonstrate how the algorithm can be used to improve MCMC algorithms for analysing models within disease mapping (Section 5.1) and space varying regression (Section 5.2), and then discuss applications in Bayesian spatial non-parametrics, Hierarchical Bayesian space-time models, Bayesian imaging and parameter estimation.

5.1 Disease Mapping

The spatial mapping of risk for a particular disease, based on observed incidence/mortality in counties or other administrative zones, is of importance for formulation and validation of aetiological hypotheses (Besag et al., 1991; Mollié, 1996). The basic model (Besag et al., 1991) is as follows. Suppose the area of interest is divided into n regions, as shown in Figure 1. Let $y_1, \ldots, y_n$ be the number of deaths from the disease of interest during the study period. The common assumption is that y_i is Poisson distributed with mean $e_i r_i$ where $\boldsymbol{e} = (e_1, \ldots, e_n)^T$ are known constant accounting for number of inhabitants etc. The relative risks $\boldsymbol{r}$ are assumed to be decomposed as $\log r_i = u_i + v_i$, where $\boldsymbol{v}$ are independent Gaussian zero mean random effects with unknown precision λ, and $\boldsymbol{u}$ is an improper GMRF with density as in (1) where κ in unknown.

A single-site MCMC algorithm for this model is easy to construct (Besag et al., 1991; Mollié, 1996). Using Gamma priors for κ and λ, the corresponding full conditionals are still Gamma and hence easy to sample from. For $\boldsymbol{v}$, one can use a component-wise random walk Metropolis step. The full conditional for the remaining term $\boldsymbol{u}$ is

$$\pi(\boldsymbol{u} \mid \text{rest}) \propto \exp\left(-\frac{1}{2}\kappa \sum_{i \sim j}(u_i - u_j)^2 + \sum_i (u_i y_i - e_i \exp(u_i + v_i))\right). \tag{8}$$

A natural choice is to use single-site update for u_i, using adaptive rejection sampling algorithm (Gilks, 1996) since $\pi(u_i \mid \text{rest}) \propto \pi(\boldsymbol{u} \mid \text{rest})$ is log-concave. Armed with the new algorithms for GMRFs, we can easily construct efficient block-sampling of $\boldsymbol{u}$. Since the full conditional for $\boldsymbol{u}$ is not a GMRF due to the exponential term $\exp(u_i)$, we approximate the exponential term in a neighbourhood around u_i as

$$\exp(u_i') \approx A_i + B_i u_i' + \frac{1}{2} C_i {u_i'}^2 \tag{9}$$

where the coefficients depends on u_i. Hence, we can use the following (non-homogeneous) GMRF as the proposal density going from $\boldsymbol{u}$ to $\boldsymbol{u}'$

$$q(\boldsymbol{u} \to \boldsymbol{u}' \mid \text{rest}) \propto \exp\left(-\frac{1}{2}\kappa \sum_{i \sim j}(u_i' - u_j')^2 + \sum_i \left(u_i' y_i - e_i \exp(v_i)(B_i u_i' + \frac{1}{2} C_i {u_i'}^2)\right)\right)$$

which is of the form of (4) with

$$Q_{ij} = \begin{cases} \sum_{k \sim i} \kappa + e_i \exp(v_i) C_i, & i = j \\ -\kappa, & i \sim j \end{cases}$$

and $b_i = y_i - e_i \exp(v_i) B_i$. The block-proposal for $\boldsymbol{u}$ is then accepted with probability $\min\{1, R\}$, where

$$R = \frac{\pi(\boldsymbol{u}' \mid \text{rest})}{\pi(\boldsymbol{u} \mid \text{rest})} \, \frac{q(\boldsymbol{u}' \to \boldsymbol{u} \mid \text{rest})}{q(\boldsymbol{u} \to \boldsymbol{u}' \mid \text{rest})}. \tag{10}$$

Note that R measures ratios of accuracy of approximations, hence will not depend heavily on n, a fact also noted by Shephard & Pitt (1997).

Let us return to the quadratic approximation (9) to the exponential. A natural (and common) choice is to use a Taylor expansion around u_i. Studying (10), it seems more important to provide an "overall" good fit to the full conditional for $\boldsymbol{u}$ in the region where $\boldsymbol{u}'$ is expected to be located, rather than a precise fit around $\boldsymbol{u}$. Hence, I will use the approximation found by

$$(A_i, B_i, C_i) = \arg\min \int_{u_i - \Delta}^{u_i + \Delta} \left(\exp(u_i') - \left[A_i + B_i u_i' + \frac{1}{2} C_i {u_i'}^2\right]\right)^2 du_i'$$

Δ	0.1	0.2	0.25	0.3	0.35	0.4
Accept-rate (%)	18.9	22.8	28.2	28.4	26.6	23.2
Step-length	7.7	7.3	7.8	7.6	7.4	7.5

TABLE 1: The average accept-rate and the average step-length (for accepted proposals) for the block-update of the $\boldsymbol{u}$ in the disease mapping example.

where Δ is a crude guess of the posterior range of u_i.

To study the (computational) performance of the GMRF part of the MCMC algorithm, I ran the model on some oral cavity cancer mortality for males in Germany (1986–1990) analysed by Knorr-Held & Raßer (2000). The map is shown in Figure 1, and the neighbours to each region i are all other regions j adjacent to i. This graph has $n = 544$ nodes with average 5.2, minimum 1 and maximum 11 neighbours. The sampling of the GMRF specified by (4) took about 0.008 seconds on a Sun Ultra 2 Model 1296 with a 296 MHz SUNW UltraSPARC-II processor. This is quite fast even though this is a small graph. The costs of the block-update of $\boldsymbol{u}$ is then about 2×0.008 seconds.

Table 1 shows the average accept-rate and step-length for accepted $\boldsymbol{u}$-proposals in $\boldsymbol{Q}$-norm (where $\boldsymbol{Q}$ is the prior precision matrix for $\boldsymbol{u}$) for varying values of Δ. The tabulated values have to be compared with the accept-rate of 16.3% and step-length 7.6 using the Taylor expansion. As the results shows, the performance of the block-update for $\boldsymbol{u}$ is quite robust for changes in Δ, and a reasonable Δ improves the accept-rate compared to using Taylor expansion, keeping the step-length approximately constant. Note that the average acceptance rate will tend to increase with increasing κ, and also the efficieny (and need) of block-updating $\boldsymbol{u}$ compared to single-site updating of $\boldsymbol{u}$.

5.2 Space Varying Regression Model

Assunção et al. (1999) study regional factor productivities and the degree of factor substitution in the Brazilan agriculture. The model is a spatial linear regression model with (smooth) spatially varying regression coeficients. MCMC is used for the inference. I will demonstrate that the new algorithm can efficiently provide a sample from the full conditional for the regression coeficients, instead of using a single-site update. The graph is more complicated than in Section 5.1.

Brazil is divided into 558 regions similar to Germany in Figure 1. Each region has a standard regression model $y_i = \boldsymbol{c}_i^T\boldsymbol{\beta}_i + \epsilon_i$ with region specific covariates $\boldsymbol{c}_i$, coeficients $\boldsymbol{\beta}_i$, and iid Gaussian noise with variance σ^2. The dimension of $\boldsymbol{\beta}_i$ is 5. Assunção et al. (1999) assume a smooth GMRF prior for the $\boldsymbol{\beta}_i$'s such that $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_j$ are similar in $\boldsymbol{\Sigma}^{-1}$ norm if $i \sim j$. Region i has all adjacent regions as neighbours. The full conditional of $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_n)^T$ is then

$$\pi(\boldsymbol{\beta} \mid \text{rest}) \propto \exp\left(-\frac{1}{2}\sum_{i\sim j} w_{ij}(\boldsymbol{\beta}_i - \boldsymbol{\beta}_j)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\beta}_i - \boldsymbol{\beta}_j) - \frac{1}{2}\sum_i (y_i - \boldsymbol{c}_i^T\boldsymbol{\beta}_i)^2/\sigma^2\right), \tag{11}$$

with known weights w_{ij}. Note that (11) defines a non-homogeneous GMRF on a quite complicated graph: if region i has m_i neighbours, then β_{ij} has $4 + 5m_i$ neighbours. This defines a graph with 2790 nodes with average 31.7, minimum 4 and maximum 74 neighbours. We can write (11) in the form of (4), with the $i + 5j$'th element of $\boldsymbol{x}$ equal to regression coefficient j in region i, which I denote as x_{ij} for short. The expressions for $\boldsymbol{Q}$ and $\boldsymbol{b}$ in (4) are obtained from (11) as

$$Q_{ij,i'j'} = \begin{cases} (\boldsymbol{\Sigma}^{-1})_{jj'} \sum_{k\sim i} w_{ik} + \sigma^{-2} c_{ij} c_{i'j'} & i = i' \\ -(\boldsymbol{\Sigma}^{-1})_{jj'} w_{ii'} & i \sim i' \end{cases} \tag{12}$$

and $b_{ij} = \sigma^{-2} y_i x_{ij}$. To obtain one sample from (11) on the same platform as in Section 5.1, takes not more than 0.6 seconds. Appendix B discuss computational details for this example.

5.3 Other applications

Bayesian non-parametrics   Heikkinen & Arjas (1998) propose a model for non-parametric Bayesian estimation of a spatial Poisson intensity. The hierarchal prior model generate a realisation as follows: first a point pattern is generated from a Poisson process, then the Voronoi tessellation and the corresponding Delaunay graph are constructed (tile $j \neq i$ is a neighbour to tile i iff they share a common edge), finally a GMRF $\boldsymbol{x}$ is added with constant values at each tile. Additionally, there are unknown parameters. I will now outline how their MCMC-algorithm can be easily improved using the new algorithms.

First I consider the full conditional for the GMRF, which is of the same form as (8),

$$\pi(\boldsymbol{x} \mid \text{rest}) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x} + \boldsymbol{c}^T(\exp(x_1), \ldots, \exp(x_n))\right) \tag{13}$$

for some $\boldsymbol{Q}$ depending on the Delaunay graph (and "the rest"), and vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ depending on "the rest". Hence the same procedure as in Section 5.1 can be used to construct a block-update step of the $\boldsymbol{x}$ conditioned on the rest. (Heikkinen & Arjas (1998) use single-site updates.) Further, when a hyperparameter or a change in the Delaunay-graph is proposed in the MCMC algorithm, the acceptance probability contains the ratio of normalisation constants including terms like $|\boldsymbol{Q}|$. A rapid evaluation of this determinant is important and is easily done using the result in Section 3.2. Hence, the local approximation of this ratio used in Heikkinen & Arjas (1998) can be replaced with the exact ratio. Finally, for very large graphs (order of $10\,000$s of tiles and above), the strategies described in Section 4 can be used, both with respect to sampling the GMRF and computing/approximating the ratio of normalisation constants.

Hierarchal Bayesian Space-time Model   Wikle et al. (1998) propose a Hierarchal Bayesian space-time model to achieve flexibility and methods for the analysis of environmental data distributed in space and time. The model consists of several stages: measurement process, large- and small-scale features, spatial structures and dynamics, priors on parameters, and hyperparameters. The processes are defined on a lattice or grid in space-time, with extensive use of GMRFs and linear relations. The model in analysed using MCMC. The proposed model is quite complicated and I will not go into (all) the details. However, the Appendix in Wikle et al. (1998) give all the full conditional densities in their model and half of them is a GMRF on a lattice. Hence, our new algorithms will be useful doing MCMC from the proposed model.

Bayesian Imaging   There are several application within imaging. One example is the high-level Bayesian model of Qian, Titterington & Chapman (1996). The task is to locate a near circular object in an electron microscopy image. The object of interest is represented by a deformable template model with one (GMRF) texture within the object, and another outside the object. Two texture-fields are defined on the hole image, but only observed inside and outside, respectively, of the object. The algorithm for sampling conditional GMRFs, can then be used to sample each of the two texture-fields (on the hole image), hence one can construct a MCMC algorithm for exploring the posterior. Qian et al. (1996) avoids this problem by not sampling from the posterior, but rather use a site-wise optimisation algorithm to locate a local mode in the posterior. Another example is the hierarchal Bayesian model of Aykroyd (1998), where a prior on the coefficients in the precision matrix for the GMRF mode for the pixels, is modelled as a GMRF. Efficient block-sampling in the MCMC algorithm is again possible.

PARAMETER ESTIMATION The rapid evaluation of the likelihood makes maximum-likelihood estimation of parameters in a GMRF possible without having to compute an approximation for the likelihood as for example in Griffith & Sone (1995) and Kiiveri (1992).

REFERENCES

ANDERSON, E., BAI, Z., BISCHOF, C., DEMMEL, J., DONGARRA, J., CROZ, J. D., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., OSTROUCHOV, S., & SORENSEN, D. (1995). *LAPACK Users' Guide*, 2 edn, Philadelphia: Society for Industrial and Applied Mathematics.

ASSUNÇÃO, J. J., GAMERMAN, D. & ASSUNÇÃO, R. M. (1999). Regional differences in factor productivities of Brazilian agriculture: a Bayesian space varying parameter approach, *Technical report*, Technical report, Statistical Laboratory, Universidade Federal do Rio de Janeiro, Brazil.

AYKROYD, R. G. (1998). Bayesian estimation for homogeneous and inhomogeneous Gaussian random fields, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(5): 533–539.

BESAG, J. & KOOPERBERG, C. (1995). On conditional and intrinsic autoregression, *Biometrika* 82(4): 733–746.

BESAG, J., YORK, J. & MOLLIÉ, A. (1991). Bayesian image restoration with two applications in spatial statistics (with discussion), *Annals of the Institute of Statistical Mathematics* 43(1): 1–59.

CARTER, C. K. & KOHN, R. (1994). On Gibbs sampling for state space models, *Biometrika* 81(3): 541–543.

CRESSIE, N. A. C. (1993). *Statistics for spatial data*, 2 edn, John Wiley, New York.

DAWID, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems, *Statistics and Computing* 2: 25–36.

DIETRICH, C. R. & NEWSAM, G. N. (1997). Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix, *SIAM Journal on Scientific Computing* 18(4): 1088–1107.

GEORGE, A. & LIU, J. W. (1981). *Computer solution of large sparse positive definite systems*, series in computational mathematics, Prentice-Hall.

GILKS, W. R. (1996). Full conditional distributions, *in* W. R. Gilks, S. Richardson & D. J. Spiegelhalter (eds), *Markov Chain Monte Carlo in Practice*, London: Chapman & Hall, pp. 75–88.

GOLUB, G. H. & VAN LOAN, C. F. (1996). *Matrix Computations*, 3 edn, Johns Hopkins University Press, Baltimore.

GRIFFITH, D. A. & SONE, A. (1995). Trade-offs associated with normalizing constants computational simplifications for estimating spatial statistical models, *Journal of Statistical Computation and Simulation* 51: 165–183.

HEIKKINEN, J. & ARJAS, E. (1998). Non-parametric Bayesian estimation of a spatial Poisson intensity, *Scandinavian Journal of Statistics* 25(3): 435–450.

JENSEN, C. S., KONG, A. & KJÆRULFF, U. (1995). Blocking-Gibbs sampling in very large probabilistic expert systems, *International Journal of Human-Computer Studies* 42: 647–666.

KIIVERI, H. T. (1992). Fitting spatial correlation models: approximating a likelihood approximation, *Australian Journal of Statistics* 34(3): 497–512.

KIIVERI, H. T. & CAMPBELL, N. A. (1989). Covariance models for lattice data, *Australian Journal of Statistics* 31(1): 62–77.

KNORR-HELD, L. & RASSER, G. (2000). Bayesian detection of clusters and discontinuities in disease maps, *Biometrics* XX(xx): xx–xx. (to appear).

LAURITZEN, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models, *Journal of the American Statistical Association* 87(420): 1098–1108.

LAVINE, M. (1998). Another look at conditionally Gaussian Markov random fields, *Bayesian Statistics: Proceedings of the Sixth International Meeting Held in Valencia (Spain)*. to appear.

LEWIS, J. G. (1982). Implementation of the Gibbs-Poole-Stockmeyer and Gibbs-king algorithms, *ACM Transactions on Mathematical Software* 8(2): 180–189.

MOLLIÉ, A. (1996). Bayesian mapping of disease, *in* W. R. Gilks, S. Richardson & D. J. Spiegelhalter (eds), *Markov Chain Monte Carlo in Practice*, London: Chapman & Hall, pp. 359–379.

OPPE, T. C., JOUBERT, W. D. & KINCAID, D. R. (1988). *NSPCG User's Guide Version 1.0 — A Package for Solving Large Sparse Linear Systems by Various Iterative Methods*, Center for Numerical Analysis, The University of Texas at Austin.

QIAN, W., TITTERINGTON, D. M. & CHAPMAN, J. N. (1996). An image analysis problem in electron microscopy, *Journal of the American Statistical Association* 91(435): 944–952.

ROBERTS, G. O. & SAHU, S. K. (1997). Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler, *Journal of the Royal Statistical Society, Series B* 59(2): 291–317.

RUE, H. & HUSBY, O. K. (1998). Identification of partly destroyed objects using deformable templates, *Statistics and Computing* 8(3): 221–228.

RUE, H. & TJELMELAND, H. (1999). Fitting Gaussian Markov random fields to Gaussian fields, *Statistics no. 16*, Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim, Norway.

SAAD, Y. (1996). *Iterative methods for sparse linear systems*, The PWS series in computational science, PWS Publishing Company.

SHEPHARD, N. & PITT, M. K. (1997). Likelihood analysis of non-Gaussian measurement time series, *Biometrika* 84(3): 653–667.

VAN DE VELDE, E. F. (1994). *Concurrent Scientific Computing*, number 16 in *TAM*, Springer–Verlag.

WIKLE, C. K., BERLINER, L. M. & CRESSIE, N. A. (1998). Hierarchical Bayesian space-time models, *Environmental and Ecological Statistics* 5(2): 117–154.

WINKLER, G. (1995). *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, Springer, Berlin.

WOOD, A. T. A. & CHAN, G. (1994). Simulation of stationary Gaussian processes in $[0, 1]^d$, *Journal of Computational and Graphical Statistics* 3(4): 409–432.

A Computational and Algorithmic Details

The basic algorithm goes in three steps.

1. Permute the indices in $\boldsymbol{Q}$ using a permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{Q}^{\text{perm}} = \boldsymbol{P}\boldsymbol{Q}\boldsymbol{P}^T$, has small bandwidth b_w.
2. Compute the band-Cholesky factorisation $\boldsymbol{Q}^{\text{perm}} = \boldsymbol{L}^T\boldsymbol{L}$.
3. Solve (using band-solvers) $\boldsymbol{L}^T\boldsymbol{x}^{\text{perm}} = \boldsymbol{z}$ where the z_i's are independent standard Gaussians, and apply the inverse permutation $\boldsymbol{x} = \boldsymbol{P}^T\boldsymbol{x}^{\text{perm}}$. Then, $\boldsymbol{x}$ is a sample from the zero mean GMRF.

Although these steps are rather basic for a numerical trained researcher, I will describe shortly each step for the statistical oriented reader, before reporting some timing results.

STEP 1. The problem of how to permute the indices in a sparse matrix to obtain a small bandwidth, is a standard problem in numerics, see for example George & Liu (1981) and Saad (1996) for an overview. Common algorithms are based on *level-sets*: a level set is defined recursively as the set of all unmarked neighbours of all the nodes of a previous level set. Starting from a clever chosen initial node in the graph, the level sets are constructed and traversed by increasing number of degree. The order by which the nodes are traversed in this algorithm determines the permutation of the indices. As always, further improvements exist which I do not discuss. Anyway, good, efficient and free implementations of such algorithms exists. I have used the algorithm as described by Lewis (1982) and the public-domain implementation available as `http://www.netlib.org/toms/582`.

STEP 2. Computing the band-Cholesky factorisation is a standard problem in numerics. The knowledge that $\boldsymbol{L}$ in $\boldsymbol{Q}^{\text{perm}} = \boldsymbol{L}^T\boldsymbol{L}$ is a lower triangular matrix with (lower-)bandwidth b_w, simply says that we do not need to compute all the zeros as in a full Cholesky factorisation. Hence, great computational savings are obtained. High-quality routines for computing the band-Cholesky factorisation are available in the public-domain `Lapack`-library written in Fortran (Anderson et al. 1995), which can be down-loaded from `http://www.netlib.org`. The appropriate routines are `dpbtf2` (level 2) or `dpbtrf` (level 3).

STEP 3. To solve $\boldsymbol{L}^T\boldsymbol{x}^{\text{perm}} = \boldsymbol{z}$ I make use of the band-linear solver `dtbsv` in the `Lapack`-library refered to in Step 2. Again, the band-version offers computational savings due to the excessive amount of known zeros, compared to a full back-substitution.

SOME TIMING RESULTS I will now present some timing results for a 100×100 lattice with varying neighbourhood when the implementation of the algorithm is run on a Sun Ultra 2 Model 1296 with a 296 MHz SUNW UltraSPARC-II processor. For a 200×100 lattice, say, the CPU will be doubled as the computational cost is linear in the largest dimension. With a 3×3 neighbourhood the algorithm used 0.92 seconds for the first sample and then produced iid samples using 0.07 seconds each. When I increased the neighbourhood to 5×5, the algorithm used 2.80 and 0.11 seconds, and for a 7×7 neighbourhood 6.00 and 0.15 seconds.

OTHER ALTERNATIVES A more state of the art package for computing bandwidth reducing permutations, is METIS (`http://www-users.cs.umn.edu/~karypis/metis/`). (For the graphs I have considered in this report, the performance is comparable.) There are other alternatives to computing the band-Cholesky factorisation (but still $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{V}^T$ for some lower sparse triangular matrix $\boldsymbol{V}$) as the best permutation (in terms of number of non-zeros in $\boldsymbol{V}$) of the nodes do not result in narrow bandwidth; the non-zeros may lie well off the diagonal. This is especially a benefit for large graphs, and there exists several packages for this purpose, like PSPASES (`http://www-users.cs.umn.edu/~mjoshi/pspases/`) and SUPERLU (`http://www.nersc.gov/~xiaoye/SuperLU/`). There are also versions of these packages for shared memory and distributed memory parallel machines, which immediate gives a parallel implementation of the sampling algorithm (with high performance).

B THE GMRFSIM-LIBRARY

`GMRFsim` is a library written in the `C`-language, with an implementation of the algorithm for sampling from a GMRF with extensions to various forms for conditional sampling, evaluations of the likelihood, and doing a MCMC block-sampling from (13). `GMRFsim` can handle both general graphs and lattices. The library is available at the URL `http://www.math.ntnu.no/~hrue/GMRFsim`.

To give an idea how easy it is to make use of the library, let us study the `C`-code needed (in condensed form) to sample from (11).

```
double *compute_b_vector() {
    for(region=0;region<n_regions;region++)
        for(indx=0;indx<5;indx++)
            b[indx+region*5] = data_precision * data_y[region] * data_x[region][indx];
    return b;
}
double Q_function_beta(int node, int nnode) {
    region  = node/5;  indx  = node  - 5*region;
    rregion = nnode/5; iindx = nnode - 5*rregion;
    if (region == rregion)
        Qvalue = sigma_inverse[indx][iindx] * accumulated_weight[region]
            + data_precision * data_x[region][indx] * data_x[region][iindx];
    else
        Qvalue = - sigma_inverse[indx][iindx] * weight[region][rregion];
    return Qvalue;
}
double *sample_beta() {
    if (!graph) graph = gmrf_g_read_graph_from_file("brasil.graph");
    cmean = compute_b_vector();
    bchol = gmrf_g_Q_solve(cmean, NULL, graph, Q_function_beta);
    gmrf_g_sim(beta, cmean, bchol, NULL, graph, Q_function_beta);
    return beta;
}
```

Some comments to the code are needed. The function `compute_b_vector` return $\boldsymbol{b}$ as $b_{ij} = \sigma^{-2} y_i x_{ij}$ as defined in Section 5.2, while `Q_function_beta` return element $Q_{ij,i'j'}$ defined in

(12). The arguments of `Q_function_beta` are either neighbours in the graph or equal. Since our internal numbering is `region`$\in \{0,\dots,557\}$ and `index`$\in \{0,\dots,4\}$ within each region, I need to compute these numbers from the node-numbers $\{0,\dots,2789\}$ in the graph.

The main function `sample_beta` proceed in the following manner. The first time `sample_beta` is called, the graph for the problem is read from the file `brasil.graph`. The file consists of lines in the following format

```
2790
0 14 5 6 7 8 9 10 11 12 13 14  1  2  3  4
[and so on]
```

The first number is the number of nodes in the graph, which are numbered as $0,\dots,n-1$. The second line says that node 0 has 14 neighbours, which are node $5,6,7,8,9,10,11$ and so on. The remaining lines in the file are similar.

After `compute_b_vector` has computed the $\boldsymbol{b}$-vector, I call `GMRFsim`-function `gmrf_g_Q_solve` to compute the conditional mean. `gmrf_g_Q_solve` solves $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b}$. The input to this function is $\boldsymbol{b}$ which is over-written by the solution $\boldsymbol{x}$. The function itself returns the band-Cholesky factorisation `bchol` of $\boldsymbol{Q}$, where the non-zero structure of $\boldsymbol{Q}$ is defined in `graph` and values by the function `Q_function_beta`. After computing the conditional mean `cmean`, I use the `GMRFsim`-function `gmrf_g_sim` to sample from the GMRF. The band-Cholesky factorisation `bchol` is known, so this step is fast. Hence, the returned `beta` contains a sample from (11).